CLAIMS

1. A method for debugging a distributed software system that includes first and second components and a coordinator that implements a desired interaction between the first and second components; each component and the said coordinator including coordination interfaces that expose events, and the debugging method comprising the steps of:

creating an event record in response to each exposed event that occurs during an execution of the distributed software system, each event record including identification of a component that generated the event and a local time stamp;

accumulating the event records into an event database; and then

displaying an evolution diagram for use by a developer in debugging the distributed software system, the evolution diagram including a graphical representation of at least a selected portion of the event database.

2. A method according to claim 1 wherein said displaying step includes displaying at least a first selected event together with the identification of the component that generated the selected event.

3. A method according to claim 1 wherein the displaying step includes presenting in the evolution diagram:

a first graphical display element representing a first component of the software system; and

a second graphical display element representing a first selected event generated by the first component;

the first and second graphical display elements being juxtaposed so as to visually indicate that the first selected event was generated by the first component.

4. A method according to claim 1 wherein the displaying step includes presenting in the evolution diagram:

a first graphical display element representing a first event;

a second graphical display element representing a second event; and a third graphical display element indicating a causal relationship between the first and second events.

5. A method according to claim 4 wherein each of the first and second graphical display elements comprises an identifier of a component that generated the corresponding event.

6. A method according to claim 4 wherein the causal relationship comprises sending a message as the first event and receiving the said message as the second event.

7. A method according to claim 6 wherein the third graphical display element comprises an arrow extending from the first graphical display element to the second graphical display element.

8. A method according to claim 4 wherein at least one of the first and second events consists of a control state change exposed at the coordination interface of one of the components.

9. A method according to claim 4 wherein at least one of the first and second events consists of a message sent from a port at the coordination interface one of the components.

10. A visual display for use by a developer in debugging a distributed or concurrent software system, the visual display comprising an evolution diagram responsive to a predetermined set of event records generated during an execution of the subject software system; each event record reflecting a corresponding software system event.

11. A visual display according to claim 10 wherein each of the event records is responsive to one of a timer tick, a data departure, a data arrival or a mode change.

12. A visual display according to claim 10 wherein the evolution diagram includes:

a first graphical display element representing a first component of the software system; and

a second graphical display element representing a first event generated by the first component;

the first and second graphical display elements being juxtaposed in the visual display so as to cue the developer that the first event was generated by the first component.

13. A visual display according to claim 12 wherein the first graphical element representing a first component of the software system comprises a generally horizontal bar and the second graphical display element comprises a generally vertical icon overlapping the horizontal bar.

14. A visual display according to claim 10 wherein the evolution diagram includes:

a first graphical display element representing a first component of the software system;

a second graphical display element representing a second component of the software system; and

a third graphical display element representing an implicit message from the first component to the second component.

15. A visual display according to claim 10 wherein the evolution diagram includes:

a first graphical display element representing a first component of the software system;

a second graphical display element representing a second component of the software system; and

a third graphical display element representing an explicit message from the first component to the second component.

16. A visual display according to claim 10 wherein the evolution diagram includes:

a first graphical display element representing a first event;

a second graphical display element representing a second event; and

a third graphical display element representing a causal relationship between the first and second events.

17. A visual display according to claim 16 wherein

the first graphical display element includes identification of a component that generated the said first event;

the first event is sending a message;

the second event is receipt of a message caused by the first event; and

the third graphical display element comprises an arrow having a tail positioned adjacent the first graphical display element and a head positioned adjacent the second graphical display element.

18. A visual display according to claim 12 wherein the first graphical display element includes indicia identifying a control state of the first component and indicating its current value.

19. A visual display according to claim 12 wherein the first graphical display element includes indicia identifying an exported variable of the first component and indicating its current value.

20. A method for debugging a distributed, hierarchical software system that includes a plurality of design levels, each design level comprising two or more components and a coordinator that implements desired interaction between the said components; each component and the coordinators with which it interacts including respective complementary coordination interfaces that expose events, and the debugging method comprising the steps of:

creating an event record in response to each exposed event that occurs during an execution of the distributed software system, each event record including identification of a component that generated the event and a local time stamp;

accumulating the event records into an event database;

partially ordering the event database based on the time stamps; and then

displaying an evolution diagram for use by a developer in debugging the distributed software system, the evolution diagram including a graphical representation of at least a selected portion of the event database.

21. A method according to claim 20 wherein said displaying step includes selecting one of the design levels and displaying an evolution diagram corresponding to the selected design level including graphical indicia including events exposed at the

coordination interfaces of components defined at the selected design level, thereby hiding subsystem interactions from view.

22. A method according to claim 21 and further comprising selecting a temporal subset of the event database for inclusion in the displayed evolution diagram thereby selectively focusing on a region of interest to the developer.

23. A method according to claim 20 wherein the execution is simulated.

24. A method according to claim 20 wherein the execution is carried out on a target hardware platform.

25. A method according to claim 20 wherein the software system is instrumented so as to generate event records at selected points in execution that are not events exposed at the coordination interface of a component.

26. A method according to claim 20 wherein the execution is carried out on a distributed, embedded target hardware platform comprising a plurality of hardware subsystems, and the event records are collected from each hardware subsystem.

27. A method according to claim 20 wherein said displaying the evolution diagram includes combining a selected sequence of events on a single component so as to form an event cluster, and then displaying the event cluster.

28. A method according to claim 20 wherein said displaying the evolution diagram includes combining a selected group of components so as to form a component cluster, and then displaying a single trace representing the component cluster.

29. A method according to claim 20 wherein said displaying the evolution diagram includes combining a selected group of state traces so as to form a state cluster, and then displaying a single trace representing the state cluster.

30. A method according to claim 20 and further comprising filtering selected events, states or components from the evolution diagram.